ALGORITMA PENCARIAN SOLUSI *GAME "MASTERMIND"*

Rudianto[1], Catherine Tobing[2], Aulia Hakim[3]

Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if14099@students.if.itb.ac.id[1], if14105@students.if.itb.ac.id[2],
if14135@students.if.itb.ac.id[3]

Abstrak

Game "Mastermind" adalah sebuah permainan sederhana yang sangat populer di masayarakat. Permainan ini dimainkan oleh 2 pemain dimana satu pemain menyusun kombinasi 4 warna dari 6 warna dan pemain yang lain mencoba menebak kombinasi warna tersebut. Pemain pertama mengajukan sebuah kombinasi dan pemain kedua akan memberikan *feedback*. Feedback yang diberikan berupa jumlah warna yang tepat dan memiliki posisi yang benar serta jumlah warna yang tepat tetapi salah posisi. Penebakan kombinasi dilakukan terus menerus sampai susunan warna terpecahkan dan akhirnya pemain bertukar posisi. Makalah ini membahas bagaimana bila algoritma *brute force* bila diterapkan untuk menemukan solusi dalam game ini. Dalam makalah ini, penulis menambahkan algoritma baru dan membandingkan kemangkusannya dengan algoritma *brute force*. Sebagai bahan telaah, di dalam makalah ini juga dijelaskan mengenai Algoritma yang cukup mangkus yang didapatkan dari situs internet.

***Kata kunci**: algoritma, mastermind, brute force*

1. Pendahuluan

Game "Mastermind" adalah permainan yang sering dimainkan untuk mengisi waktu, bahkan varian dari permainan ini sudah biasa dimainkan di kelas oleh mahasiswa. Game ini pada dasarnya adalah sebuah permainan tebak-tebakan yang disertai umpan balik dari pemberi tebakan. Seringkali para pemain sering hanya asal menebak dengan sedikit berpikir. Pada dasarnya game ini dapat diselesaikan dengan mudah dengan algoritma Brute Force akan tetapi algoritma ini membutuhkan kemungkinan langkah yang sangat banyak. Ada beberapa kemungkinan algoritma lain yang dapat dilakukan antara lain dengan cara mengurangi jumlah kemungkinan warna menjadi hanya sejumlah letak yang tersedia. Algoritma yang berusaha dibuat oleh penulis adalah dengan cara mengeliminasi jumlah kemungkinan warna menjadi sebanyak jumlah letak yang tersedia pada awalnya sehingga proses *brute force* berikutnya menjadi lebih singkat. Akan tetapi, algoritma ini juga masih kurang mangkus, oleh karena itu penulis berusaha mencari algoritma lain yang lebih mangkus. Akhirnya ditemukan sebuah algoritma yang cukup mangkus untuk menyelesaikan persoalan game MasterMind ini. Algoritma ini menyederhanakan persoalan dengan cara membatasi jumlah tebakan pertama yang mungkin, kemudian berdasarkan feedback yang didapatkan dieliminasi beberapa kemungkinan yang ada. Algoritma ini dilakukan berulang kali dengan tebakan yang berbeda. Dengan menggunakan algoritma ini, banyaknya proses penebakan yang harus dilakukan, dalam *worst case*, hanya berjumlah 5 kali. Algoritma yang terakhir ini agak sulit ditampilkan pseudo codenya oleh karena itu penulis hanya akan menjelaskan secara kata-kata berdasarkan referensi yang ada.

2. Peraturan game Master Mind

Peraturan yang digunakan adalah peraturan standar yaitu tersedia 6 warna yang kemudian dipilih 4 warna (boleh kurang) yang akan ditempatkan dalam suatu susunan 4 elemen. Kombinasi setiap susunan boleh berulang serta setiap kali penebakan harus lengkap seluruh posisi dan tidak ada pembatasan jumlah penebakan. Penebak soal tidak diberitahu apakah posisi peletakan warna sudah tepat atau belum. Feedback yang diberikan oleh pemberi soal hanya berupa jumlah warna yang benar dan berada pada posisi yang tepat, yang diwakilkan dengan bulatan putih serta jumlah warna yang benar dengan posisi yang salah yang diwakilkan dengan bulatan hitam.

3. Pencarian Solusi dengan Algoritma Brute Force

Sesuai dengan prinsip *brute force*, untuk mengerjakan persoalan ini, penjawab soal perlu untuk mencari semua kombinasi warna dan letaknya yang mungkin. Tentunya proses ini akan memakan waktu yang cukup lama karena harus memperhitungkan semua kemungkinan dengan cara permutasi. Kemungkinan terburuknya adalah kita harus mencari sebanyak permutasi jumlah warna

diambil sebanyak jumlah tempat. Kompleksitas algoritmanya adalah O(nPm) dengan n adalah jumlah warna yang dimiliki dan m adalah jumlah letak yang tersedia. Penyelesaian persoalan ini tidak memperhatikan *feedback* dari pemberi soal.

Contoh bila kita ingin menebak 4 susunan warna (merah, kuning, biru, hijau) dengan mengambil kemungkinan yang ada di warna pelangi (merah, jingga, kuning, hijau, biru, nila, ungu) maka kita harus mencoba satu persatu kemungkinan :

Kemungkinan	Feedback
merah-merah-merah-merah	○
merah-merah-merah-jingga	○
merah-merah-merah-kuning	○●
merah-merah-merah-hijau	○●
…. dst ….	
hijau-kuning-merah-biru	○●●● *)

*) bulatan di posisi awal tidak menandakan bahwa posisi awal sudah ditempati oleh warna yang benar, tetapi hanya menandakan ada 1 warna yang telah menempati tempat yang benar.

4. Pemecahan dengan Beberapa Algoritma Lain

Algoritma pertama yang ingin coba diaplikasikan untuk memecahkan persoalan *Master Mind* adalah dengan menambahkan *feedback* dalam proses pencarian solusi sehingga jumlah kemungkinan dapat dikurangi.

Pada awalnya buatlah sebuah prosedur yang menerima masukan 4 warna serta mengembalikan keluaran jumlah bulatan putih dan jumlah bulatan merah.

```
procedure Master_Mind
    (input A,B,C,D : string)

Deklarasi
    solusi : array [1..4] of string
    warna : array [1..6] of string
Algoritma
    solusi[1] ← A
    solusi[2] ← B
    solusi[3] ← C
    solusi[4] ← D
    while (nbBulatPutih ≠ 4) do
        {generate kemungkinan (A,B,C,D) }
        {kemudian diassign ke array
         coba_jawab}
        {bagian ini yang akan diubah}
        {kemudian panggil Cek_Jawaban}
        Cek_Jawaban(input warna[1],warna[2]
                    Input warna[3],warna[4])
    endwhile
{berikutnya prosedur yang tidak diubah
karena sesuai dengan aturan Master_Mind}

procedure Cek_Jawaban
    (input tebakan : array [1..4] of string
     output nbBulatHitam : integer
     output nbBulatPutih : integer)

Deklarasi
    i : integer
    nbBulat : integer
Algoritma
    nbBulat ?  min(merah_di_solusi,
merah_di_tebakan) + min(jingga_di_solusi,
jingga_di_tebakan) + min(kuning_di_solusi,
kuning_di_tebakan) + min(hijau_di_solusi,
hijau_di_tebakan) + min(biru_di_solusi,
biru_di_tebakan) + min(nila_di_solusi,
nila_di_tebakan)
    for i=1 to 4 do
       if (coba_jawab[i]=solusi[i]) then
                nbBulatPutih+=1
       endif
    endfor
    nbBulatHitam ? nbBulat – nbBulatPutih

{ fungsi min(argumen1, argumen2)
  mengembalikan argumen1 jika
  arguman1 < argumen2, dan sebaliknya }
```

Algoritma Brute Force biasa tentu saja masih sangat tidak efisien. Langkah berikutnya, kita dapat menambahkan *feedback* sebagai salah satu bahan pertimbangan. Pada bagian generate kemungkinan kita masukkan kombinasi dengan setiap warna yang sama untuk setiap komponennya (misal : merah-merah-merah-merah). Secara otomatis itu akan menghasilkan warna yang digunakan walaupun belum diketahui tempatnya dari 4 bulatan (baik putih maupun hitam) yang muncul. Setelah menemukan jumlah masing-masing warna yang digunakan, kita dapat kembali melakukan metode brute force dengan jumlah kemungkinan yang lebih kecil yaitu hanya 4 warna untuk dicocokkan di 4 tempat. Jumlah kemungkinan yang perlu dilakukan menjadi bersisa (6+(4P4)). Kompleksitas algoritmanya menjadi O(m!) dimana m adalah jumlah letak yang tersedia.

```
procedure Master_Mind
    (input A,B,C,D : string)

Deklarasi
    solusi : array [1..4] of string
    warna : array [1..6] of string
    temp_solusi : array [1..4] of string
    i,j,k : integer
Algoritma
    k←1;
    solusi[1] ←  A
    solusi[2] ←  B
    solusi[3] ←  C
    solusi[4] ←  D
    for i=1 to 6 do
      for j=1 to 4 do
         coba_jawab[j] ← warna[i]
         Cek_Jawaban(coba_jawab[])
         if (nbBulatPutih≠0||nbBulatHitam≠0)
         then
              temp_solusi[k]← warna[i]
              k+=1
         endif
      endfor
    endfor
{selanjutnya digenerate semua kemungkinan
(A,B,C,D) yang tersisa, A,B,C,D kombinasi
 dari array temp_solusi}
    while (nbBulatPutih ≠ 4) do
        {generate kemungkinan (A,B,C,D) }
        {kemudian diassign ke array
         coba_jawab}
        {bagian ini yang akan diubah}
        {kemudian panggil Cek_Jawaban}
        Cek_Jawaban(input warna[1],warna[2]
                    Input warna[3],warna[4])
    endwhile
```

Algoritma Brute Force dengan penambahan itu masih dirasakan kurang efektif. Untuk itu diperlukan algoritma lain yang lebih efektif.

Berikutnya akan dijelaskan sedikit mengenai algoritma yang penulis temukan sebagai salah satu algoritma termangkus. Pada dasarnya, hanya ada 5 macam tebakan pertama, dalam hal ini kita masukkan umpamakan sebagai : BBBB, BBBC, BBCC, BBCG, and BCRG. Setiap variable berisi warna yang unik. Susunan ini dapat berlaku untuk warna apapun asalkan jumlah total warna adalah 6. Kemudian kemungkinan *feedback* dari pembuat soal hanya 14 yaitu :
(0) Tanpa tanda
(1) 1 BulatHitam
(2) 1 BulatPutih
(3) 2 BulatHitam
(4) 1 BulatPutih, 1 BulatHitam
(5) 2 BulatPutih
(6) 3 BulatHitam
(7) 1 BulatPutih, 2 BulatHitam
(8) 2 BulatPutih, 1 BulatHitam
(9) 3 BulatPutih
(10) 4 BulatHitam
(11) 1 BulatPutih, 3 BulatHitam
(12) 2 BulatPutih, 2 BulatHitam
(13) 4 BulatPutih [Solusi ditemukan!!!! ^_^]

Kemudian berdasarkan *feedback*, kemungkinan solusi berkurang dari sebelumnya (6+4!) menjadi sebagai berikut :

	Tebakan Pertama				
Feed-back:	BBBB	BBBC	BBCC	BBCG	BCGR
0	625	256	256	81	16
1	0	308	256	276	152
2	500	317	256	182	108
3	0	61	96	222	312
4	0	156	208	230	252
5	150	123	114	105	96
6	0	0	16	44	136
7	0	27	36	84	132
8	0	24	32	40	48
9	20	20	20	20	20
10	0	0	1	2	9
11	0	0	0	4	8
12	0	3	4	5	6
13	1	1	1	1	1

Berdasarkan tabel, langkah pertama yang paling optimum adalah BBCC karena pada *worst case* menyisakan 256 calon solusi.

Dengan cara mengulangi procedure di atas untuk langkah kedua dan selanjutnya dengan jumlah kemungkinan yang tersisa, kita dapat memperoleh solusi yang kita inginkan (dijelaskan dalam tabel solusi pada referensi). Kemungkinan terburuk solusi akan dicapai setelah 5 kali pengulangan prosedure (5 kali melakukan tebakan).

Pada dasarnya konsep algoritma ini adalah mengeliminasi setiap state yang telah tidak mungkin menjadi solusi secara bertahap berdasarkan *feedback* yang diberikan oleh pemberi soal.

Berdasarkan referensi, ada satu algoritma yang paling mangkus mengenai persoalan mind master ini yang ditemukan oleh Kenji Koyama and Tony W. Lai pada tahun 1993. Algoritma tersebut jika dihitung akan menghasilkan nilai rata-rata sebesar 5625/1296 langkah.

5. Kesimpulan

Master Mind merupakan sebuah game yang sangat menarik. Berbagai penyelesaian dapat dilakukan. Algoritma yang paling sederhana adalah *brute force* dengan cara mencoba semua kemungkinan yang ada tanpa mempertimbangkan *feedback* yang akan didapat. Menggunakan metode ini jumlah kemungkinan terburuk yang harus dicek adalah nPm, dengan m adalah jumlah letak dan n adalah jumlah warna.

Algoritma yang sedikit lebih maju adalah dengan cara memasukkan kombinasi setiap warna yang sama untuk setiap komponen dan secara otomatis akan menghasilkan buletan sebanyak m (baik hitam maupun putih) Kemudian dari 4 warna yang didapat itu dilakukan kembali metode *brute force*. Dengan ini kemungkinan terburuk yang harus dicek menjadi

(n + m!) dimana n=jumlah warna dan m=jumlah letak.

Algoritma yang cukup mangkus ditemukan oleh penulis pada sebuah situs, yang dapat menemukan solusi dengan kasus terburuk hanya dengan melakukan lima kali tebakan. Proses yang dilakukan adalah mengeliminasi setiap state yang telah tidak mungkin menjadi solusi berdasarkan *feedback* dan tebakan selanjutnya dipilih berdasarkan sisa kemungkinan yang masih ada.

Ada banyak algoritma yang dapat digunakan untuk menyelesaikan persoalan Master Mind ini dan untuk menemukan algoritma yang paling mangkus adalah suatu tantangan yang menarik.

5. Daftar Pustaka

1. http://www.tnelson.demon.co.uk/mastermind/

2. http://www.tnelson.demon.co.uk/mastermind/table.html

3. Munir Rinaldi. 2006. *Strategi Algoritmik IF2251*, Bandung : Penerbit ITB.